Final

A Multi-Mission Deep Space Telecommunications Analysis Tool: The Telecom Forecaster Predictor

Ramona H. Tung
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
MS: 161-260
Pasadena, CA 91109
818-354-5068
Ramona.H.Tung@jpl.nasa.gov

Kevin K. Tong
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
MS: 161-260
Pasadena, CA 91109
818-354-7967
Kevin.K.Tong@jpl.nasa.gov

Abstract—The Telecom Forecaster Predictor (TFP) fulfills the need for a powerful, easily adaptable, multi-mission telecommunications analysis tool. The software is used in operations by five JPL missions, and adaptations exist for 13 missions.

Built upon MATLAB®[1], the TFP combines both multi-mission and mission-specific models to predict performance. Basing new mission adaptations on existing ones while sharing common models reduces development time and the number of modeling errors. Link configurations are specified through a user-friendly GUI, and navigation inputs are provided through standard JPL NAIF SPICE[2] SPK trajectory and CK attitude files, which the TFP can read directly.

Current work involves the development of a mission planning tool based on the TFP architecture. The objective is to create an evolvable telecommunications analysis tool to support missions from start to finish. The modular architecture of the TFP is a solid framework upon which other multi-mission tools can be modeled.

TABLE OF CONTENTS

1. INTRODUCTION[3]

In the past, telecommunication analysts at the Jet Propulsion Laboratory (JPL) would build their own tools for mission support. These tools would differ in architecture, user interface, and software basis[4], even though their primary purpose was the same. Data and formulations common to all missions were not shared which resulted in a duplication of effort. Modeling differences and errors often went uncorrected because there was no convenient baseline for comparison. Configuration management, if it existed, was very loose. There was a need for an easily adaptable telecommunications tool for supporting a wide variety of Deep Space missions, and the TFP was created to fill this void.

The TFP analysis tool, which is described in detail in [2], is built upon the popular MATLAB computing environment. Users can customize their own TFP sessions without changing official versions. Inputs are entered through the TFP's main Graphical User Interface (GUI), which has a characteristic look-and-feel independent of mission adaptation. A sample TFP GUI is shown in **Figure 1**. The building blocks of the TFP are models (specialized MATLAB scripts) which are organized in a logical fashion. Model hierarchy is traceable through an automatically created model tree, and individual models can be examined using the TFP's model editing tool. After execution, outputs are viewable in plot or tabular[5] form, and design control tables provide snapshots of link performance at a single time point.

The TFP model libraries are analogous to MATLAB toolboxes. Multi-mission models reside in an area accessible to all missions, whereas mission-specific models are stored in individual mission areas. Models are easily modified or replaced which allows great flexibility. Existing mission models are often reused or used as templates by new missions which accelerates development.

MATLAB is used extensively by the scientific community, and provides a familiar environment for TFP users. There are many reasons for choosing MATLAB as the engine for

[1] MATLAB® is a trademark of TheMathWorks, Inc.

[2] The Spacecraft, Planets, Instruments, C-Matrix, Events (SPICE) data system is a software system consisting of a set of standard database formats (referred to as kernels or SPICE files) and a set of library functions which allow users to retrieve data from these files. This software standard was introduced by JPL's Navigation Ancillary Information Facility (NAIF) Group and is gaining acceptance outside of JPL. For more info on SPICE, see [1].

[3] 0-7803-5846-5/00/$10.00 © 2000 IEEE

[4] Previous tools were based on Microsoft® Excel or PERL.

[5] Time-stamped data can be saved in comma-separated variable (CSV) form for export to other applications like Microsoft® Excel.

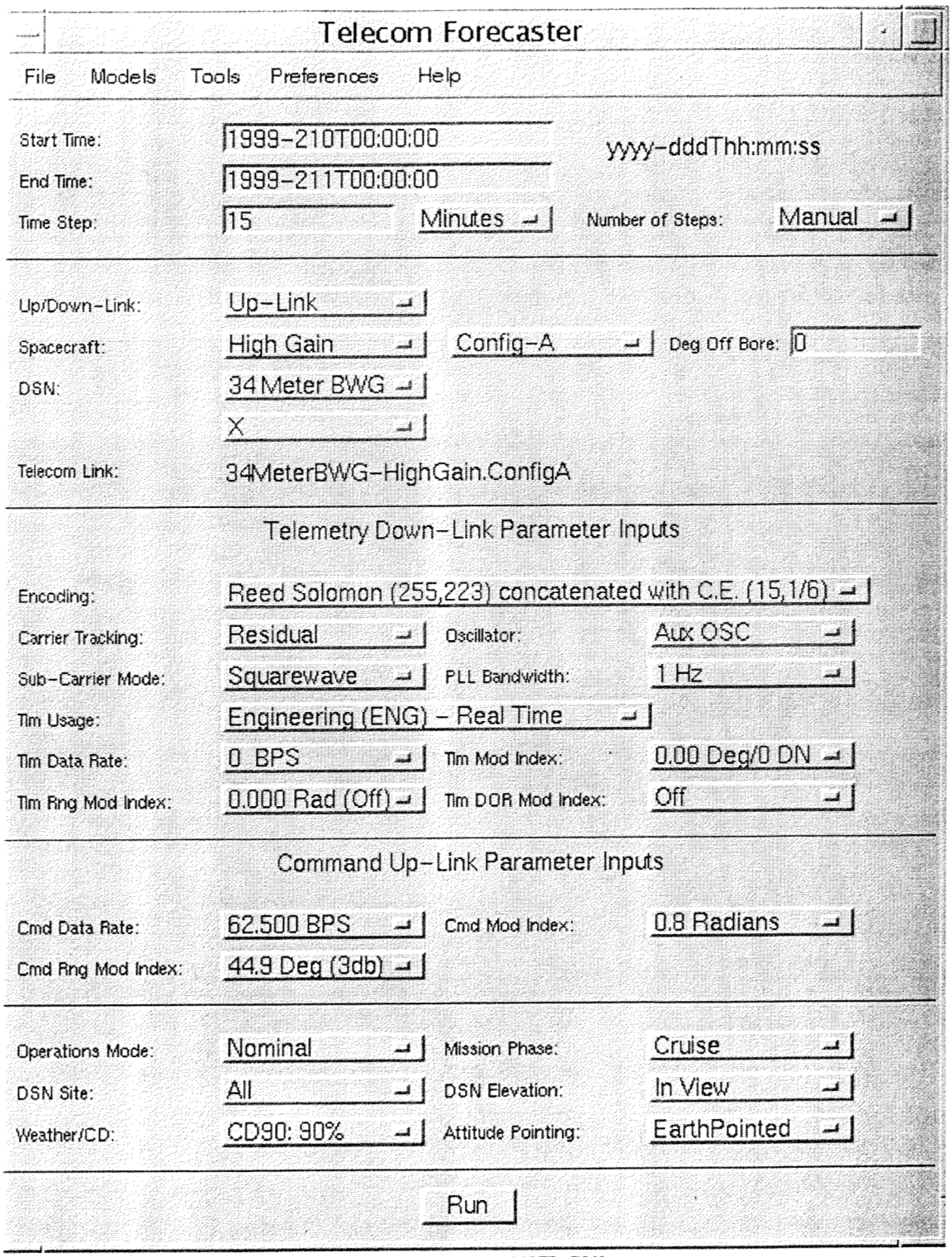


Figure 1 A Typical TFP GUI

the TFP. MATLAB code is portable across all platforms supported by MATLAB, which currently include Sun and HP workstations and PC Win9x/NT[6]. The software offers easy to use GUI builder tools, C/C++ and FORTRAN interfaces, excellent graphing capabilities, and powerful and proven built-in functions. Since MATLAB is a matrix-based computing platform, it easily manages the time-based vector calculations performed by the TFP.

[6] Macintosh support ceased with MATLAB version 5.1. The TFP is not available for the Macintosh.

2. GUI FEATURES

The TFP GUI is a frame that allows several independent tools to work together seamlessly to form an integrated simulation environment. The TFP software features dynamic script formation, model tracing, a model editing and viewing tool, GUI button linkage, and state saving capability. In addition, it has a special C/C++/FORTRAN interface for importing navigation data.[7]

Dynamic Script Formation

Dynamic script formation refers to the automatic generation of simulation code while parameters are set in the GUI. The TFP internally forms the main execution script from the current GUI state (which determines the top-level model that controls the simulation) and the model libraries. There is a finite, but different, set of top-level models for each mission, so use of a dynamic main script is cleaner and more efficient than enumerating all possible choices with logical switch statements. It also relieves the programmer of main script changes if additional GUI inputs are needed.

Model Tracing

The model tree is derived from the top-level model determined by the GUI state. It is built on demand, and allows the user to visually trace through the simulation sequentially at a high level. When one model calls (or "imports") another model, it is similar to calling a subroutine: it executes the code in the imported model, then returns to the calling model. The model tree depicts the full hierarchy of calling and called models.

A sample model tree is shown in **Figure 2**. The tree is built in the following manner. First, the top-level model, 34BWG-HighGain.ConfigA, is listed. All models that it imports directly are listed in sequential order below it. If any of these models imports additional models, an extension arrow (>) is placed next to the model name. In the example, BWGDirEirp imports additional models. Following the arrow shows another listing, beginning with BWGDirEirp followed by all the models it imports directly. This procedure is followed sequentially until the end of each branch is reached.

When the Models menu is invoked, only the models that the top-level model imports directly are shown. The model tree is traversed one level at a time by clicking on the extension arrows. Clicking on a model name that does not have an extension arrow automatically loads a copy of the model into editpar, which is the TFP's model editing tool.

Model Editing and Viewing

Editpar is the TFP's tool for editing and viewing models. An editpar window containing a sample top-level model is shown in **Figure 3**. The tool encourages the programmer to write equations and algorithms in a standard format, keeping code short and simple. Support code that performs more complicated functions is off-loaded to scripts and functions created with a text editor. In this manner, higher-level algorithms and simulation flow can be viewed using editpar. If desired, detailed support code can be examined using any text editor.

GUI Button Linkage

The design of the TFP GUI allows buttons to be linked together to perform limited constraint checking. In other words, choosing a selection in one button can affect the list of choices of one or more other buttons. The GUI allows an unlimited level of linkages[8], but has a built in mechanism that prevents an infinite loop if more than one level of linkages is used.

State Saving

The save state function is a powerful feature of the TFP GUI. When selected, it saves the current state of the GUI to a GUI state file (GSF) and allows a user to restore the simulation configuration in the future. In addition, a batch script is automatically generated. This batch script can be invoked from the MATLAB command line, performs the same analysis, and saves the results in a MATLAB data file.

If the GUI creation software is modified after a GSF is saved, a warning will be issued when a user tries to restore the GUI state. The TFP will restore as much of the saved GUI state as it can, then it will check the restored state against the saved state and highlight any differences. A summary of the saved GUI state appears in the MATLAB command window (and in a log file) for reference. After the user corrects any discrepancies, the GSF should be saved again.

3. MODEL ARCHITECTURE

Models are the building blocks of the TFP. They are created using the editpar tool, and translated into MATLAB script files that are combined to perform the requested analysis. The TFP core contains a library of common models, scripts, and functions, which is accessible to all missions. The common model library is similar to a MATLAB toolbox. It contains validated Deep Space Network (DSN) models[9], useful link analysis computations,

[7] MATLAB provides a general C/C++/FORTRAN interface. It is possible to write routines that allow the TFP to read from a wide variety of input sources.

[8] That is, the selection in one button can affect the selections in another button, which in turn may affect other buttons, and so forth.

[9] Currently, the TFP contains the latest available DSN data. In the future, it will automatically import DSN data from an official repository (as soon as the DSN creates this database). The links to the database are already in place.

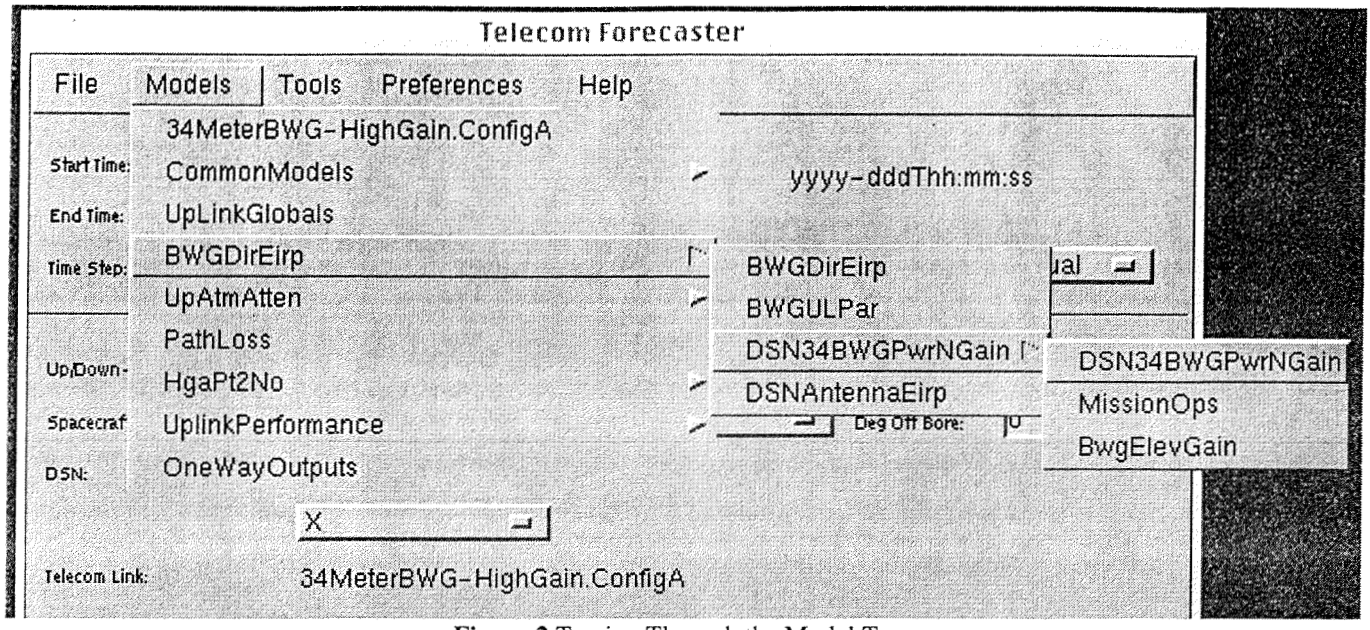


Figure 2 Tracing Through the Model Tree

and general utilities. While common model use is not required, it is highly recommended because the models contain up-to-date information, and have proven to be robust.

Top-level models (which are mission-specific) control the flow of the simulation. Typically, there are three types of top-level models: uplink, downlink, and two-way[10]. Top-level models of the same type have the same internal structure and differ only by the specific antenna models they import. Furthermore, this common structure is maintained regardless of the mission adaptation; that is, *all* mission top-level models of a given type have the same general architecture. For instance, **Figure 3** depicts a typical uplink top-level model showing an uplink from the 34-meter beam waveguide (BWG) antennas[11] to a spacecraft's high gain antenna (HGA). Note that it imports the BWG transmit model and the HGA receive model. The top-level model describing an uplink from the 34-meter standard (STD) antennas[12] to the spacecraft's low gain antenna (LGA) would have exactly the same structure except it would import the 34-meter STD transmit model and the LGA receive model. These transmit and receive models are usually interchangeable because the models were written with a generalized method for accounting gains and losses.

Dividing the TFP simulation into models organizes the code. As mentioned in the previous section on GUI design, models can be nested, and the model tree displays a visual hierarchy of the execution of the code. When designing the model architecture, calculations and information common to all missions were identified and placed into the common model area. All other calculations were placed in mission-specific models. Within models, the use of standardized parameter names was adopted to simplify and accelerate the adaptation process.

The "typical" Deep Space telecommunications analysis was examined and logically divided into high-level models. Common components include a space loss (or path loss) model, atmospheric loss models, and ground receive/ transmit models. Examples of mission-specific models are spacecraft transmit/receive models, uplink and downlink parameter initialization models, and output generation models. Each high–level model was further subdivided into common and mission-specific components. Related computations were naturally grouped into blocks that could be easily replaced if missions decided to use alternative models.

A simple philosophy governs the organization of the analysis. Initially, external data (e.g., NAIF SPICE data) is brought in and geometric, time-based parameters such as range, azimuth, elevation, degrees-off-boresight (or cone), and clock are pre-calculated for all time points. Constants for conversions (e.g., speed of light and Boltzman's constant) and internal indicator flags are also initialized up front. Subsequently, mission-specific constants are defined. Two models consolidate most mission-specific parameters so that if their values need updating (such as when a new adaptation is being generated), they are easily found. After mission-specific constants are defined, a typical link analysis is performed, and outputs are generated. The different output products of the TFP are described in the next section.

[10] Two-way involves a simultaneous uplink and downlink at the same ground station. Internally, the run is organized into an uplink followed by a downlink.

[11] The 34-meter BWG antennas are in the DSN.

[12] The 34-meter STD antennas are also in the DSN.

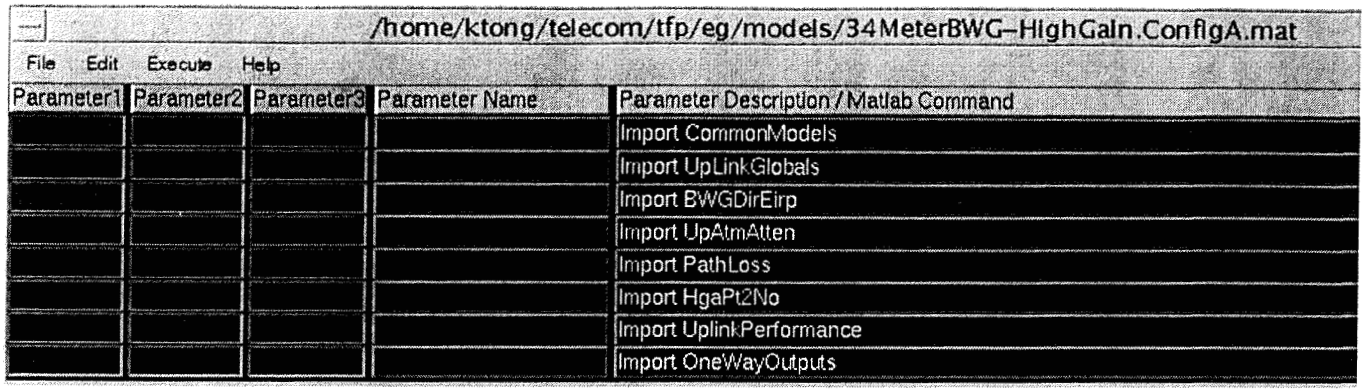


Figure 3 A Sample Uplink Top-level Model in Editpar

4. Data Visualization

Formatted Data Files

The TFP generates a standard set of output files for the analyst's convenience. These include Design Control Tables (DCTs) and a Doppler predicts table. A DCT provides a detailed snapshot of link performance at a single time point. It is complete accounting of the gains and losses in the telecommunications link. Received power levels in the carrier, data, and ranging channels are computed then compared with thresholds. The TFP produces uplink, downlink, and ranging DCTs. A Doppler predicts table is also returned to help DSN ground equipment compensate for the expected Doppler shift.

It is easy to create additional data files. All major variables used in a simulation are available after a TFP run, and MATLAB file-I/O commands are modeled after C, making it straightforward to write scripts that produce customized outputs. As an example, the TFP provides a specially formatted data rate capability file to one of the JPL missions.

QuickPlots and Reports

QuickPlots provides a convenient method for viewing data of interest in the workspace. The QuickPlots menu contains a list of typical variables of interest to telecommunications analysts[13]. Users can select up to 9 variables to either plot on the screen[14] or save in a Report file. Reports are comma separated variable (CSV) files that contain the time-stamped values of the selected variables. CSV files can be imported into standard spreadsheet programs like Microsoft Excel.

Workspace Visualization Tool

The Workspace Visualization Tool is a graphical analysis tool that can quickly access and analyze time-based data in the workspace after a run. This tool is GUI-based and is called from a pull-down menu in the TFP GUI. It is intended for users who are not familiar with MATLAB's plotting commands, and includes a predefined set of plot routines and filters that are commonly used by telecommunications analysts.

5. Conclusions

Built upon MATLAB, the TFP is a powerful, easily adaptable, multi-mission analysis tool for Deep Space telecommunications analysis. It has proven to be dependable and robust through its extensive use for mission support at JPL. Common information is shared between missions, which limits redundancy and improves accuracy. The TFP also offers many useful options for viewing and processing results.

The TFP model architecture has been generalized such that new adaptations are usually modeled after existing adaptations. This allows them to inherit the full advantages of previous adaptations. Adherence to current TFP modeling philosophies and naming conventions drastically reduces development time. Changing variable names and bookkeeping strategies for gains and losses requires significant customization and is not recommended unless absolutely necessary. Fortunately, the architecture conforms to generally accepted conventions where ever possible.

The TFP is adaptable to support different mission phases, but currently it is primarily an operations tool. Immediate plans involve the development of a mission planning tool based on the TFP architecture. The main objective is to create an evolvable telecom analysis tool that supports a mission from start to finish (design to operations), which yields tremendous savings in time and cost. Possible future endeavors include generalizing the TFP to analyze communications between any two bodies, such as spacecraft-to-spacecraft or lander-to-orbiter.

One can envision basing other types of multi-mission analysis tools on the TFP architecture. Several telecommunications tools based on the TFP architecture are currently in use at JPL. These include the Derived Channel Processor, which allows comparison of predictions with actual data, the Unified Telecom Predictor, which provides a

[13] The items on this menu are easily customizable.

[14] Plots on the screen can be printed or saved in a wide variety of formats using the MATLAB *print* command.

specific batch mode interface to the TFP for the DSN, and the Radio Frequency Interference (RFI) Analyzer, which can be used to study the interference of signals from multiple spacecraft. Though the TFP models are telecommunications-specific, the TFP software architecture is general enough to manage many types of engineering simulations.

ACKNOWLEDGMENTS

The TFP and this paper were a collaborative effort of both authors. The authors would like to acknowledge Bruno Calanche and Jeff Steinman for their early contributions to the development of the TFP.

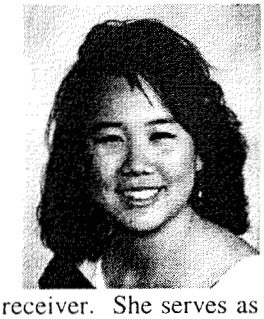
The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

REFERENCES

[1] Chuck Acton (at JPL), *The SPICE Description Paper.*

[2] Kevin K. Tong and Ramona H. Tung, *Telecom Forecaster Predictor (TFP) User's Guide and Reference*, JPL IOM 33110-99-010, October 1, 1999.



Ramona H. Tung is a telecommunications analyst at the Jet Propulsion Laboratory. Prior to designing and implementing the model architecture of the TFP, she helped design, build, and test the programmable maximum-likelihood convolutional decoder used by the DSN, and supported the development of the Block V digital receiver. She serves as a telecommunications analyst and consultant for several of JPL's missions. She received her BS and MS in Electrical Engineering and Computer Science from MIT in 1992 and 1994 respectively and has been working at JPL since 1991.

Kevin K. Tong *is a telecommunications software engineer at the Jet Propulsion Laboratory. Prior to designing the software and GUI architecture of the TFP, he helped develop and implement the data compression software used by the Galileo S-Band mission. Before coming to JPL in 1991, he was involved in image processing research at the Hughes Aircraft Company. Kevin received his BS and MS in Electrical and Computer Engineering from the University of California, Los Angeles and the University of Southern California in 1984 and 1986 respectively.*